IN THE SPECIFICATION

Please replace the paragraph beginning at page 1, line 18 through page 2, line 2, with the following rewritten paragraph:

Conventionally, even in the case in which a processor had a circuit configuration corresponding to a mov instruction, for example (mov Rn, Rm which moves the contents of register Rm into the register Rn), it was not possible for the programmer to generate a high-level programming language program using the mov instruction unless a compiler accommodates the mov instruction. In such cases, in spite of the fact that the architecture of the processor being used by the programmer accommodates the mov instruction, [[the]] it is not possibly to simply code "mov Rn, Rm". Instead, in order to achieve the same function it was necessary to code the program so as to use the lw (load word) and sw (store word) instructions. That is, even though the processor itself has a circuit configuration corresponding to the mov instruction, the program must be executed so as to use a plurality of instructions such as lw and sw or the like.

Please replace the paragraph at page 2, lines 3-7, with the following rewritten paragraph:

In the case of using a compiler that does not accommodate instructions that cannot be achieved using with a combination of existing instructions, such as a multimedia instruction, in which parallel processing is done [[of]] on a plurality of data with a single instruction, the programmer cannot make use of the characteristic functions of the processor.

Please replace the paragraph at page 2, lines 12-17, with the following rewritten paragraph:

Conventionally, a method of having the compiler accommodate the characteristic specifications (processor architecture and instruction set) was that of using intrinsics function, these being ~~function~~ functions provided beforehand as part of the processing system for the programming language, and being built-in functions provided for the purpose of compiling instruction code of a high-level language characteristic to the target processor into machine language.

Please replace the section heading at page 8, line 16, as follows:

BRIEF DESCRIPTION OF THE ~~DRAWING~~ DRAWINGS

Please replace the paragraph at page 10, lines 14-24, with the following rewritten paragraph:

Because the __reg_dest and __reg_src are identifier names for dummy arguments, it is possible, in accordance with the ISO/JIS C language standard, to code them as arbitrary identifier names. The dummy arguments are used in the definition of the function, and these are replaced by actual arguments when the function is called. When defining an intrinsics function, by using these special reserved words, information with regard to the operation of the mov instruction is transmitted to the compiler. The __reg_dest is the register destination operand, and __reg_src is the register source operand. Although in the first embodiment, __asm, __reg_dest, and __reg_src are specifically used as reserved words, [[and]] any word or phrase other than those reserved in the ISO/JIS C language standard can be used as an intrinsics function defining reserved word.

Please replace the paragraph at page 12, lines 8-13, with the following rewritten paragraph:

Fig. 1 is a drawing showing ~~the of a~~ an intermediate code generation type of compiler. As shown in Fig. 1, a compiler according to the first embodiment comprises a character string ~~interpreter~~ analyzer 11, a syntax analyzer 12, an intermediate code generator 13, an intermediate code optimizer 14, a code generator 15a, a code optimizer 16a, a code output unit 17a, and an intrinsics function information database 18.

Please replace the paragraph at page 12, lines 14-17, with the following rewritten paragraph:

The compiler 10a first performs character string interpretation by using the character string ~~interpreter~~ analyzer 11. The character string ~~interpreter~~ analyzer 11 performs processing so as to divide instructions coded in the source program 1 into minimum units called tokens.

Please replace the paragraph at page 12, lines 18-22, with the following rewritten paragraph:

Next, the syntax analyzer 12 performs syntax analysis. At the syntax analyzer 12, instructions that have been divided into tokens, which are minimum units having meanings such as variables, symbols, and the like, divided by the immediately previous character string ~~interpreter~~ analyzer 11, performs a check of syntax to see whether these are syntactically suitable as defined by each particular language.

Please replace the paragraph at page 13, lines 17-21, with the following rewritten paragraph:

Fig. 2 is a drawing showing the configuration of a type of compiler which does not generate intermediate code. As shown in Fig. 2, a compiler 10b generates object code 3 from a source program 1a via a character string ~~interpreter~~ analyzer 11, a syntax analyzer 12, a code generator 15b, a code optimizer 16b, and finally a code output unit 17b.

Please replace the paragraph at page 23, lines 14-17, with the following rewritten paragraph:

3) In the case in which the first argument of an intrinsics function is a destination, by changing the function definition, it requires no necessity to change the part that uses [[a]] an instruction (for example, mov instruction) having the same name as the name of the intrinsics function (for example, mov function).